Datorsystem

Tentamen

2012-03-17

Instruktioner

Samtliga svar skall vara motiverade och läsbara. Eventuella tabeller, illustrationer och beräkningar som används för att nå svaret ska också finnas med i lösningen. Ett svar måste vara läsligt för att kunna bedömas. Samtliga antaganden skall anges i samband med uppgiftens lösning. **Ofullständigt motiverade svar kan inte ge full poäng!**

Alla svar ska skrivas på rättningsblad, svar kan alltså inte lämnas i tentahäftet. För del A kan flera frågor besvaras på samma sida, men för del B ska samtliga frågor besvaras på en egen sida.

Hjälpmedel

Inga hjälpmedel är tillåtna.

Svar på frågor

En av kursens examinatorer kommer vid minst två tillfällen att besöka alla tentasalarna för att svara på eventuellt uppkomna frågor.

Rättning och betygsskala

Denna tentamen består av två delar: A och B. Del A kan maximalt ge betyget E, högre betyg nås i del B. För att del B ska rättas måste betyget E ha nåtts på del A. Del A och del B har varsin poängskala enligt tabellen nedan.

Del A				**Del B**			
F	Fx	E		D	C	B	A
0-7	8-10	11-16		6-9	10-15	16-20	21-24

Resultatet av tentamen kommer att meddelas senast 9 april 2012. Den som får betyget Fx måste ta kontakt med kursens examinator senast 16 april 2011. Komplettering för betyg Fx kommer att vara skriftlig, eller skriftlig samt muntlig, beroende på examinators bedömning.

Lycka till!

Del A

1. (a) Konvertera det hexadecimala talet $A5_{16}$ till ett tiobitars binärt tal. (1 A-poäng)

Lösning: $A5_{16} = 00\ 1010\ 0101_2$. Eftersom svaret ska vara tio bitar får vi ”padda” med nollor på de två mest signifikanta bitarna.

(b) Konvertera det hexadecimala talet $-A5_{16}$ till ett tiobitars binärt tal på tvåkomplementsform.

Lösning: Från uppgift 1a har vi den binära representationen av $A5_{16}$, och kan med hjälp av den ta fram tvåkomplementsformen för att få det negativa talet.

Steg 1: Invertera det binära talet:

!	0	0	1	0	1	0	0	1	0	1
=	1	1	0	1	0	1	1	0	1	0

Steg 2: Addera 1 till det inverterade talet

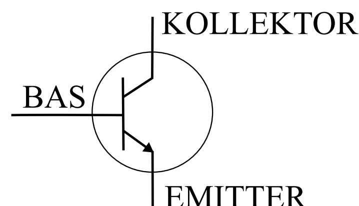
	1	1	0	1	0	1	1	0	1	0
+										1
=	1	1	0	1	0	1	1	0	1	1

Svar: $-A5_{16}$ är $11\ 0101\ 1011_2$ i tvåkomplementsform.

(1 A-poäng)

2. (a) Ett av de stora stegen i datorns utveckling togs när elektronrören ersattes av de snabbare och mer robusta transistorerna. Förklara transistorns funktion. (1 A-poäng)

Lösning: En transistor kan fungera som en strömbrytare som kontrolleras av en styrström. Transistorn har tre ben: bas, kollektor och emitter.



I kollektorn finns en insignal och i emittern en utsignal. Om en (tillräckligt stor) styrström leds in på basen kommer insignalen från kollektorn gå ut från emittern, annars kommer signalen att vara 0 ut från emittern oavsett signalen från kollektorn. I moderna datorsystem används strömmarna för att representera ettor och nollor. En ström representerar en etta och avsaknad av en ström representerar en nolla. På det sättet kan transistorer användas för att utföra aritmetiska operationer samt som en beståndsdel i DRAM-minnen.

(b) Vilken logisk operation används för att maska fram angivna bitar ur en bitsträng? Illustrera med ett exempel. (1 A-poäng)

Lösning: För att maska ut delar av en bitsträng används den logiska operationen OCH. Som exempel kan vi maska ut bit 5 ur följande bitsträng: 0110 0101.

	0	1	1	0	0	1	0	1
OCH	0	0	1	0	0	0	0	0
=	0	0	1	0	0	0	0	0

I vårt resultat finns nu bara en enda etta, den vi maskade ut.

3. En av många egenskaper som kan skilja mellan olika processorarkitekturer är om processorarkitekturen är little- eller big-endian.

(a) Vad innebär begreppet endianness? (1 A-poäng)

Lösning: Endianness anger i vilken följd information hanteras av processorn; alltså om den minst signifikanta biten hämtas först eller sist. I en byte-adresserad processor kommer det vara ordningen av bytes som bestäms av vilken endian som används.

(b) Processorn ska spara värdet 0x12345678 på adress 0x100 i primärminnet. Hur kommer informationen att vara lagrad på en little-endian processor respektive på en big-endian processor i ett byte-addresserat datorsystem? (1 A-poäng)

Lösning: En byte är åtta bitar. Eftersom 0x12345678 är 32 bitar kommer fyra adresser att användas, med en byte på varje adress.

Adress	**Big endian**	**Little endian**
0x100	12	78
0x101	34	56
0x102	56	34
0x103	78	12

4. Under datorns historia har processorarkitekturen utökats med olika tekniker för att förbättra processorns prestanda. En av de viktigaste teknikerna är pipelineing.

(a) Förklara principen bakom pipelineing samt hur en processor som stöder pipelineing bearbetar instruktioner. (1 A-poäng)

Lösning: Principen bakom pipelining är att låta processorn exekvera flera instruktioner samtidigt. Man gör det genom att dela upp exekveringen av en instruktion i flera steg. Processorn kan då parallellt hantera lika många instruktioner som steg i pipelinen, varje steg hanterar en instruktion åt gången.

(b) Hinder för pipelinens prestanda kallas pipeline hazards. Ange två exempel på pipeline hazards samt förklara vad de innebär. (1 A-poäng)

Lösning:

Data hazard Kan uppstå när en instruktion i pipelinen är beroende av data från en annan instruktion. Ett exempel på detta är när en instruktion behöver läsa data som en tidigare instruktion ännu inte hunnit beräkna, så kallad read after write (RAW).

Structural hazard Kan uppstå när två instruktioner samtidigt är på steg som kräver tillgång till en specifik resurs, men där resursen bara kan hantera en förfrågan åt gången. Ett exempel på detta är om två instruktioner är på steg som kräver tillgång till minnet, till exempel Fetch Instruction och Write Operands.

Control (branch) hazard Kan uppstå i samband med en villkorad branch-instruktion. Processorn behöver lägga till instruktioner i pipeline, men kan råka lägga till en instruktion som inte kommer utföras på grund av branchen.

5. En av processorns beståndsdelar är registren. Processorns register kan separeras i två kategorier; general purpose- samt kontrollregister.

   (a) Vad är skillnaden mellan general purpose- och kontrollregister? (1 A-poäng)

   Lösning: Ett general purpose-register är ett register som programmeraren har full tillgång till för att läsa och skriva data till. Kontrollregister kan enbart kan läsas och innehåller någon typ av statusinformation, som till exempel vilken instruktion som exekveras eller programräknaren.

   (b) Förklara syftet som följande kontrollregister har: Program Counter (PC) samt Instruction Register (IR) (1 A-poäng)

   Lösning:

   Program Counter innehåller adressen till den instruktion som processorn ska exekvera härnäst.

   Instruction Register innehåller maskinkoden för den instruktion som processorn exekverar.

6. Operativsystemet fungerar som ett mjukvarulager mellan datorsystemets programmerare/-slutanvändare och dess hårdvara. Välj ut de, i ditt tycke, tre viktigaste tjänsterna som operativsystemet ger åt programmeraren/slutanvändaren. Motivera dina val utförligt! (2 A-poäng)

Lösning:

Schemaläggning av processer Utan schemaläggning av processer skulle datorsystemets multitasking bli omständig. Just funktionen att användaren kan starta flera processer utan att behöva hantera när dessa program ska exekveras är grunden för hur moderna persondatorsystem fungerar.

Minneshantering Majoriteten av dagens datorsystem inkluderar virtuellt minne, vilket gör det enklare för en programmerare att skriva programkod eftersom man slipper ta hänsyn till att fragmenterat minne och liknande. Det är också en användbar tjänst för en användare av ett datorsystem eftersom man kan starta fler processer än vad som får plats i primärminnet.

Systembibliotek och drivrutiner Som programmerare underlättar det att kunna använda operativsystemets bibliotek för fönsterhantering, filhantering och liknande, istället för att själv behöva hantera exempelvis skrivning och läsning till hårddisk.

Det finns ett flertal andra giltiga svar här, så länge motiveringen är tydlig och korrekt. Till exempel gränssnitt (grafiska och textbaserade), filsystem med flera.

7. I TCP/IP-modellens transportlager ligger protokollen UDP och TCP. Båda protokollen erbjuder portar samt en checksumma för att kontrollera att den överförda informationen är korrekt.

   TCP erbjuder också, utöver vad som nämns ovan, pålitlig leverans samt att data levereras i korrekt ordning. Förklara hur TCP tekniskt kan erbjuda dessa båda tjänster. (2 A-poäng)

Lösning: Att data levereras i rätt ordning sker genom att varje paket innehåller ett sekvensnummer, där sekvensnumret ökar för varje nytt paket.

Att leveransen är pålitlig betyder att de paket som skickas över en existerande uppkoppling kommer fram, och det görs genom att mottagaren skickar en bekräftelse på varje paket den tar emot. Om avsändaren inte får bekräftelse på att ett paket har tagits emot skickar den paketet igen.

8. De senaste 30 åren har IPv4 varit det dominerande nätverksprotokollet på internet. När IPv4 skapades såg inte internet ut som det gör idag, med en nätverksansluten enhet i var mans hand. Därför designades protokollet med tanken att 2,4 miljarder adresser räcker långt nog, och idag har vi en brist på tillgängliga IPv4-adresser

   En teknik som har utvecklats för att dryga ut adresserna är Network Address Translation (NAT). Under 1990-talet utvecklades dessutom en ny version av IP-protokollet: IPv6.

   (a) Hur används NAT för att utöka mängden användbara IPv4-adresser? (1 A-poäng)

Lösning: NAT översätter från en IP-adress till en annan. Med hjälp av den funktionen kan ett internt nätverk dela på en enda publik IPv4-adress som sedan översätts till en privat IPv4-adress för varje inkopplad enhet.

När en enhet i det interna nätverket vill skicka en förfrågan till en enhetet utanför det privata nätverket måste paketets avsändaradress ändras till en publik IP-adress innan informationen lämnar nätverket, detta görs av en brandvägg i nätverket.

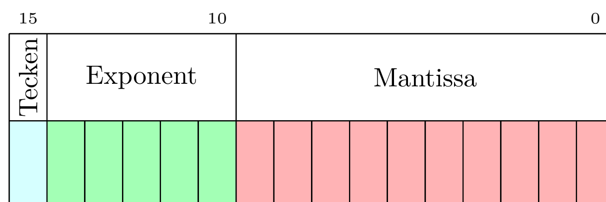
Då svaret på förfrågan kommer tillbaka till nätverket tar brandväggen emot paketet och ändrar mottagaradressen från den publika adressen till den adressen för den enheten som gjorde förfrågan.

(b) Vilken är skillnaden mellan ett IPv6- och ett IPv4-paket som gör att IPv6 har en större adressmängd? (1 A-poäng)

Lösning: I headern för ett IPv6-paket används 128 bitar för att ange mottagarens och sändarens adress, medan IPv4 använder 32 bitar i samma syfte. Fler tillgänliga bitar innebär ett större antal möjliga adresser.

Del B

1. Flyttal är datorsystemets sätt att representera stora, små och rationella tal. I nedanstående uppgift används flyttalsstandarden IEEE 754 för 16-bitars flyttal. Figur 1 illustrerar hur flyttalet lagras binärt.



Figur 1: Ett 16-bitars flyttal enligt IEEE 754

Värdet på ett 16-bitars flyttal kan enligt IEEE 754 beräknas med formeln:

$$v = (-1)^{teckenbit} * 2^{exponent-15} * (1,mantissa)_2$$

(a) Beräkna operationen A * B. Svara med ett 16-bitars flyttal enligt IEEE 754.
Flyttal A: 0 10010 1011010000
Flyttal B: 0 10110 1011000000 (2 B-poäng)

Lösning: Först beräknar vi exponenterna för de båda talen. A: $10010_2 = 18$, och B: $10110_2 = 22$. Från formeln ska vi också subtrahera 15 från båda talen: 18-15=3 och 22-15=7.

Eftersom det är multiplikation av flyttal kan vi använda följande formel: $(2^x \times a) \times (2^y \times b) = 2^{x+y} \times (a \times b)$. Vi behöver därmed inte göra något mer åt exponenterna även om de skiljer sig åt, utan kan addera dem rakt av. 3+7=10, och vi har därmed början på exponenten i vårt svar.

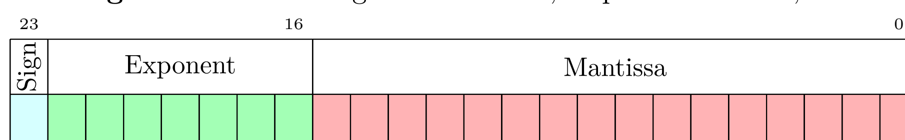
Nästa steg blir att multiplicera mantissorna med varandra, efter att man lagt på den etta som finns i formeln.
1,101101
$*$1,1011
Vi multiplicerar mantissorna från talen A och B och får följande:

					1,	1	0	1	1	0	1
				*			1,	1	0	1	1
					1	1	0	1	1	0	1
				1	1	0	1	1	0	1	
			0	0	0	0	0	0	0		
		1	1	0	1	1	0	1			
+	1	1	0	1	1	0	1				
1	0,	1	1	0	1	1	1	1	1	1	1

Mantissan är just nu 10,1101111111, men den måste normeras för att vara på rätt format enligt formeln. Vi får skifta ett steg åt höger och får följande: 10,1101111111 » 1,01101111111. Inledande 1, plockas bort eftersom det ingår i formeln, och mantissan är nu 01101111111. Problemet är att det är 11 bitar, och i mantissan finns bara plats för 10. Det blir ett overflow; den sista biten måste tas bort. Den resulterande mantissan på 10 bitar blir: 0110111111.

> Eftersom vi skiftade mantissan så att den blev mindre måste vi öka värdet av exponenten, och får då 10+1=11. Sist måste vi, enligt formeln, addera 15: 11+15=26_2 = 11010_2.
>
> Resultatet av multiplikationen av de båda talen som ett 16-bitars flyttal är alltså 0 11010 0110111111.

(b) IEEE 754 finns definierat för flyttal av storleken 8, 16, 32, 64 samt 128 bitar. Ponera att du är arbetar med ett datorsystem som använder den (något udda) ordlängden 24 bitar. Hur många bitar avsätter du åt tecken, exponent och mantissa? Hur kommer formeln för att beräkna flyttalets värde se ut för ditt 24-bitarsformat? Motivera dina svar! (2 B-poäng)

> **Lösning:** Vald bitindelning: Tecken: 1 bit, Exponent: 7 bitar, Mantissa: 16 bitar.
>
> 
>
>
> Jämfört med 16-bitars flyttal, även kallat half-precision, är det fler bitar för mantissan i relation till exponenten. Anledningen till det är att exponenten redan nu är två bitar längre än vid half-precision, och det ger ett tillräckligt stort spann av tal. Att då lägga fler bitar till mantissan ger en större möjlighet till precision även vid något större tal, eftersom det måste vara längre tal för att få overflow. Teckenbiten är fortfarande bara en bit, eftersom det fortfarande bara finns två lägen att välja mellan.
>
> Eftersom vi använder fler bitar för exponenten måste vi räkna ut ett nytt *bias*, vilket vi gör genom att ta $2^6 - 1 = 63$. 2^6 kommer från att vi har sju bitar för exponenten, så det är alltså värdet av den mest signifikanta biten i exponenten. Formeln blir då följande:
>
> $$v = (-1)^{teckenbit} * 2^{exponent-63} * (1, mantissa)_2$$

2. Datorsystemet har en minneshierarki med flera olika typer av minne med olika prestanda och storlek. En av de snabbaste typerna av minne är cacheminnet.

   (a) Antag att vi har ett datorsystem med ett cacheminne som har följande egenskaper:

Storlek:	256 bytes
Radlängd:	8 byte
Associativitet:	4-vägs
Skrivpolicy:	Write back
Ersättningspolicy:	Least Recently Used (LRU)

Tabell 1: Cacheminnets egenskaper

Följande minimala assemblyprogram kommer att utföras av datorsystemets processor:

```
movia   r8 ,  0x0800222F
ldw     r10 , 0x35(r8)    # 0x08002264
stw     r11 , 0xC1(r8)    # 0x080022F0
ldw     r12 , 0xB7(r8)    # 0x080022E6
sth     r16 , 0x1B1(r8)   # 0x080023E0
ldh     r13 , 0x31(r8)    # 0x08002266
ldb     r14 , −0xE(r8)    # 0x08002221
ldw     r15 , 0x75(r8)    # 0x080022A4
ldb     r16 , 0x36(r8)    # 0x80002265
```

För varje minnesreferens, ange om instruktionen kommer att resultera i en cache-träff eller en cache-miss samt på vilken rad och i vilken mängd i cacheminnet som informationen kommer att sparas. (2 B-poäng)

Lösning: Först måste vi räkna ut bitindelningen av adressen för att kunna adressera cacheminnet utifrån primärminnesadressen. På varje rad finns 8 bytes, alltså behövs 3 bitar (000 - 111) för att adressera en specifik byte på en rad.

När det gäller raderna måste vi börja med att räkna ut hur många rader som finns i cacheminnet. Eftersom det är ett fyrvägsassociativt cacheminne finns fyra mängder per rad, och det måste tas med i beräkningen: 256 bytes totalt, och fyra mängder per rad med åtta bytes per rad ger 256/(8∗4)=256/32=8 rader. I cacheminnet finns alltså 8 rader, och då behövs 3 bitar för att adressera en enskild rad (000 - 111).

Bitindelning:

31 … 6	5 4 3	2 1 0
Tag	Rad	Byte

ldw r10, 0x35(r8) # 0x08002264 08002264_{16} = 000 1000 0000 0000 0010 0010 0110 0100_2. De sex minst signifikanta bitarna är 100 100$_2$, och ur dem får vi att raden blir 100 och byten blir också 100. Vi söker alltså data på rad 100, men eftersom cacheminnet är tomt får vi en **cachemiss**. På rad 100, mängd 0 läggs data från adresse 0x08002260 till 0x08002267 in.

stw r11, 0xC1(r8) # 0x080022F0 Vi såg ovan att bara de sex minst signifikanta bitarna behöv för att adressera rad och byte. Därför kommer vi fortsättningsvis bara omvandla dessa bitar från hex till binärt, och sedan jämföra Tag decimalt vid behov.

$F0_{16}$ = 11 11 0 000$_2$, vilket ger rad 110. Rad 110 är tom, alltså får vi en **cachemiss**. På rad 110 läggs 0x080022F0 - 0x080022F7 in i mängd 0.

ldw r12, 0xB7(r8) # 0x080022E6 $E6_{16}$ = 11 10 0 110$_2$. På rad 100 finns redan data i mängd 0, och vi måste jämföra Tag. Redan de två minst signifikanta bitarna i Tag är olika, och vi får därmed en **cachemiss**. Därför läggs 0x080022E0 - 0x080022E7 in på rad 100, mängd 1.

sth r16, 0x1B1(r8) # 0x080023E0 $E0_{16}$ = 11 10 0 000$_2$. Raden är återigen 100, och vi får jämföra Tag. Mängd 0 ser vi skiljer redan på de två minst

signifikanta bitarna, medan mängd 1 skiljer sig senare: Tag för mängd 1: 0x080022C, Tag för sökt data: 0x080023C. Vi får en **cachemiss** och lägger in 0x080023E0 - 0x080023E7 på rad 100, mängd 2.

ldh r13, 0x31(r8) # 0x08002266 $66_{16} = 01\ 10\ 0\ 110_2$. Ännu en gång får vi rad 100 och måste jämföra Tag, vilket ger en **träff** i mängd 0.

ldb r14, -0xE(r8) # 0x08002221 $21_{16} = 00\ 10\ 0\ 001_2$. Vi hamnar på rad 100 och måste jämföra Tag, men hittar inget matchande data. Vi får en **cachemiss** och lägger in 0x08002220 - 0x08002227 i mängd 3.

ldw r15, 0x75(r8) # 0x080022A4 $A4_{16} = 10\ 10\ 0\ 100_2$. Ännu en gång hamnar vi på rad 100 och får jämföra Tag. Återigen finns inte det data vi söker, vilket ger en cachemiss. Alla fyra mängder är fulla, och vi måste kasta ut data från en av mängderna.

Vi har policyn Least Recently Used, så vi ska kasta ut data från den mängd vars data användes för längst tid sedan. Det är mängd 1 som är Least Recently Used, alltså kastar vi ut data därifrån och lägger in 0x080022A0 - 0x080022A7 på rad 100, mängd 1.

ldb r16, 0x36(r8) # 0x80002265 $65_{16} = 01\ 10\ 0\ 101_2$. Raden är 100, och där finns data i samtliga mängder. Med hjälp av Tag finner vi sökt data i mängd 0, och har alltså en **cacheträff**.

(b) Cacheminnen kan användas för att förbättra prestandan för minnesreferenser som uppvisar tidslokalitet och/eller rumslokalitet.

(a) Vad innebär begreppen tidslokalitet och rumslokalitet?

Lösning:

Rumslokalitet innebär att man utgår från att ett program inom kort kommer att referera till data på en adress i närheten av den nuvarande referensen.

Tidslokalitet innebär att man utgår från att samma data som refereras inom kort kommer att användas igen.

(b) Hur kan man anpassa ett cacheminnes egenskaper för att fungera bättre för minnesreferenser som uppvisar tidslokalitet respektive minnesreferenser som uppvisar rumslokalitet?

Lösning: Man kan tänka sig flera olika varianter för att anpassa ett cacheminne för tidslokalitet:

- Man kan använda ett **direktmappat** cacheminne med en kort **radlängd** för att få maximalt antal rader vilket innebär att så många referenser som möjligt ska få plats i minnet.

- Man kan också tänka sig ett **associativt** cacheminne med **radlängd** enligt exemplet ovan. På det sättet kommer referenser inte att skriva över varandra utan fylla upp minnet.
- Ett ytterliggare alternativ är att använda ett **flervägsassociativt minne**, eftersom man då kan spara data längre även om nya referenser hamnar på samma rad.

För att optimera ett cacheminne för rumslokalitet bör man maximera cacheminnets **radlängd**, så att man kan läsa in data runtomkring den referens som görs. Längre rader med fler bytes ger större möjlighet att utnyttja rumslokalitet i minnesreferenserna.

(2 B-poäng)

3. Processorns arbete illustreras med den tillståndsmaskin som kallas Fetch/Execute-cykeln.

   (a) Vilka är de fyra olika tillstånden som ingår i Fetch/Execute-cykeln, beskriv vad som händer i varje tillstånd. Mellan vilka tillstånd kan körningen växla? (2 B-poäng)

Lösning:

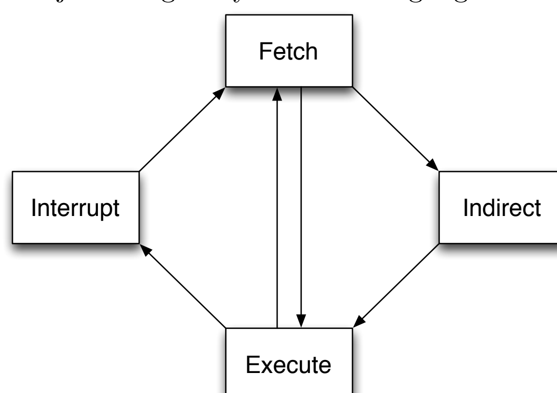
Fetch I tillståndet Fetch hämtas en instruktion från minnet till processorn.

Execute I tillståndet Execute läses OP-koden av och motsvarande instruktion exekveras.

Interrupt I tillståndet Interrupt sparas det nuvarande värdet på programräknaren undan och processorn börjar istället exekvera en avbrottshanterare.

Indirect I en processor med stöd för indirekt adressering behövs ytterliggare data inhämtas för att hämta värdet från den adress som hämtas i första Fetch-steget.

I följande figur syns hur övergångarna mellan de olika tillstånden kan ske:



   (b) För varje steg i tillståndsmaskinen; ange vilka delar av processorn och enheter i datorsystemet som bearbetar varje steg. (2 B-poäng)

Lösning:

Tillstånd	Processordelar	Övriga enheter
Fetch	PC, MAR, MBR, IR, CU	Primärminne, systembuss
Execute	Register, CU, ALU	Primärminne, systembuss, I/O-enheter
Interrupt	PC, MAR, MBR, CU	Primärminne, systembuss
Indirect	MAR, MBR, CU	Primärminne, systembuss

Det man får tänka på är att för Execute-tillståndet varierar inblandade enheter beroende på instruktion och operander.

4. (a) Moderna datorsystem instrueras genom att program som skrivs i ett högnivåspråk utförs av datorsystemets enheter. Datorsystemetet förstår dock endast instruktioner i binärt format. Redogör för hur högnivåspråk omvandlas till maskininstruktioner i binärt format. (2 B-poäng)

Lösning: Första steget är att översätta högnivåspråket till ett lågnivåspråk, ett steg kallat kompilering. Språket som används för resultatet kan, men måste inte vara, Assembler eller ett Assembler-liknande språk.

Andra steget är att ta resultatet från kompileringen och assemblera det. Det innebär att kod i det mellanliggande språket översätts till binära maskininstruktioner, så kallad objektkod. I det här steget ingår att utifrån den givna instruktionen slå upp rätt OP-kod, välja rätt instruktionsformat samt lägga in operander såsom register och immediate-värden. Både kompilatorn och assemblatorn kan också försöka optimera programkoden för att få ett bättre slutresultat.

Sista steget är länkningen. Där sätts program skrivna i flera filer ihop, och adresser till anropade systembibliotek läggs in.

(b) Idag finns två olika huvudtyper av processorarkitekturer; RISC och CISC. Vilka är filosofierna bakom de båda arkitekturerna? Beskriv användningsområden för de båda processorarkitekturerna och motivera varför arkitekturen passar för detta användningsområde. (2 B-poäng)

Lösning:

Reduced Instruction Set Computer Filosofin bakom RISC är att förenkla instruktionsuppsättninge på olika sätt. Med en enklare instruktionsuppsättning menas bland annat att man använder färre instruktionsformat, har samma längd på alla instruktioner, använder enklare adressering och att varje instruktion ska ta en maskincykel.

Instruktionerna ska också vara av typen register-till-register, vilket innebär att läsningar och skrivningar till minnet enbart sker med speciella instruktioner. Även registeruppsättningen är annorlunda, då fler register används

jämfört med CISC-processorer.

Utgångspunkten är att man med en enklare instruktionsuppsättning och fler förenklar jobbet för kompilatorerna, som därmed kan göra sitt jobb bättre. Även pipelining kan förenklas med en enklare instruktionsuppsättning.

Användningsområden för RISC-arkitekturen är ofta mindre och mindre kraftfulla datorsystem som exempelvis handhållna enheter eller inbyggda system. Ofta har dessa enheter mindre eller ingen cache.

Complex Instruction Set Computer Filosofin bakom CISC är att tillåta både register- och minnesoperander, samt att försöka matcha mer avancerade högnivåspråk med mer avancerade instruktioner i processorn. En annan del är att ha variabelt långa instruktioner för att få tätare kod, vilket kan spara plats i minnnet. CISC är ett något vidare begrepp än RISC, eftersom CISC är en term som skapades som motpol till RISC.

Användningsområden för CISC är i regel mer kraftfulla datorsystem, exempelvis har arkitekturerna x86 och x86_64 drag av CISC.

5. Moderna operativsystem hanterar datorsystemets minnet åt användaren. En vanlig teknik som används är virtuellt minne.

   (a) Vad innebär virtuellt minne? Vad skiljer det virtuella minnet från det reella minnet? (2 B-poäng)

Lösning: Virtuellt minne är ett sätt att utöka mängden tillgängligt minne genom att kombinera utrymmet i primärminnet med utrymme på sekundärminnet. Det är också vanligt att arkitekturer med virtuellt minne tilldelar varje process ett eget virtuellt, sammanhängande utrymme som ofta också börjar på adress 0.

Det som skiljer det virtuella minnet från det reella minnet är att det reella, eller fysiska, minnet är begränsat av antalet platser i primärminnets kretsar, medan det virtuella minnet egentligen inte har några begränsningar så länge det finns plats på sekundärminnet. En annan skillnad är de adresser som används. Eftersom varje process kan tilldelas ett eget virtuellt minne kan flera processer använda samma minnesadress, men utan att de mappar mot samma reella adress. En reell adress kan aldrig mappa mot flera områden i primärminnet.

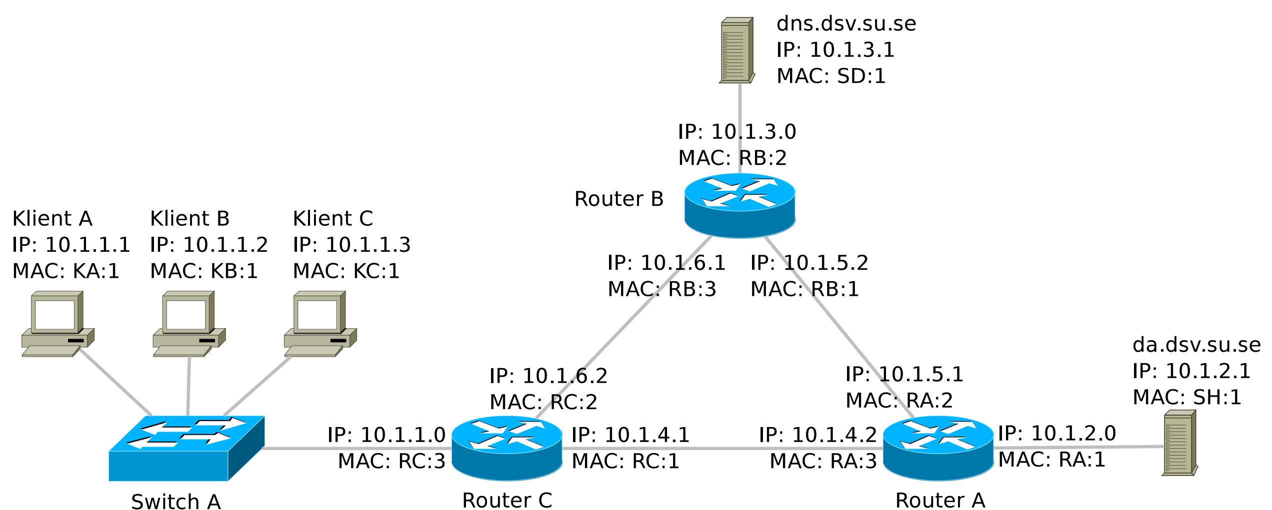
   (b) Ange vilka enheter i datorsystemet som är involverade i virtuellminneshanteringen. För varje enhet, förklara dess roll i hanteringen. (2 B-poäng)

Lösning: Om virtuellt minne används behövs en enhet som hanterar översättningen från virtuella till reella adresser. Den enheten kallas Memory Management Unit (MMU), och den ligger mellan processorn och primärminnet. Processorn använder virtuella adresser i sina instruktioner, och dessa adresser går sedan till MMUn som översätter dem till fysiska adresser innan de går vidare till primärminnet.

Även primär- och sekundärminnet är involverade i hanteringen av virtuellt minne. På primärminnet lagras allting i första hand, och på primärminnet lagras den information som inte får plats i primärminnet. Sekundärminnet är därmed ett krav för virtuellt minne, eftersom det annars inte skulle gå att uttöka minnesmängden.

Om cacheminne används för att snabba upp anrop till primärminnet är även det involverat i virtuellminneshanteringen. Cacheminnet kan dock sitta både före och efter MMUn, och kan därför arbeta med virtuella eller fysiska adresser beroende på placeringen av MMUn.

6. I Figur 2 visas ett exempel på ett mindre nätverk med flera subnät och flera anslutna nätverksenheter.



Figur 2: Ett mindre nätverk

(a) Illustrera trafiken som går genom nätverket när Klient A vill ansluta till http://da.dsv.su.se. Utgå från att Klient A är konfigurerad att kommunicera via Router C och har adressen till DNS-servern, men i övrigt inte vet något om nätverket. Om flera alternativa vägar existerar; redogör för alla möjliga alternativ. (2 B-poäng)

Lösning: Klient A kommer först att behöva hämta IP-adressen till da.dsv.su.se, och sedan göra själva anropet till webbservern. Trafiken kommer då troligen gå enligt följande: Klient A → Switch A → Router C → Router → dns.dsv.su.se → Router B → Router C → Switch A → Klient A.

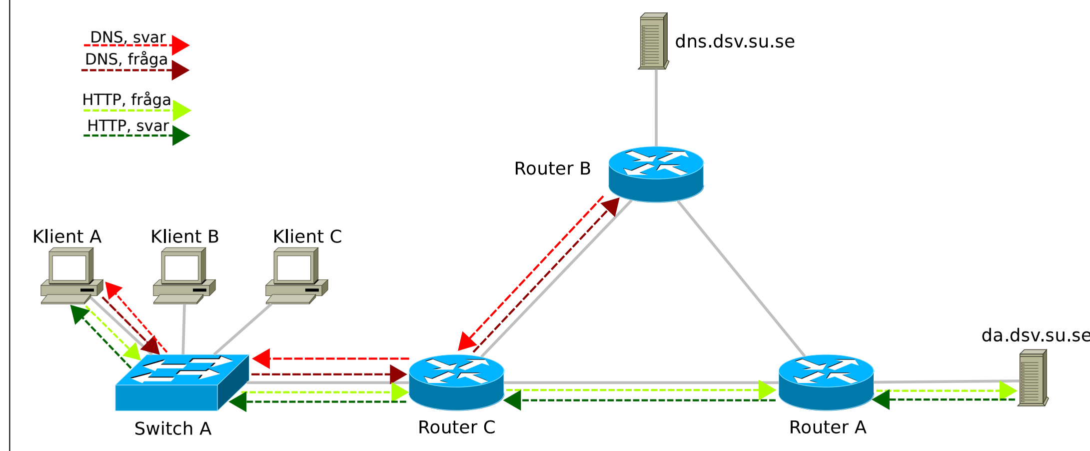
Alternativa vägar är följande: Router C → Router A → Router C för vägen till DNS-servern, och sedan Router B → Router A → Router C på vägen tillbaka.

När Klient A väl vet IP-adressen till webbservern kommer Klient A skicka en HTTP-request som mest troligt kommer ta följande väg: Klient A → Switch A →

Router C → Router A → da.dsv.su.se. Alternativa vägar är Router C → Router B → Router A.

När webbservern svarar kommer trafiken troligen gå da.dsv.su.se → Router A → Router C → Switch A → Klient A, med alternativa vägen Router A → Router B → Router C.

Bilden nedan illustrerar de båda flödena, men tar inte med de alternativa vägarna.



(b) Upprätta en tabell som visar HTTP-förfrågans väg genom nätverket. För varje steg i nätverket, ange paketets IP-adress för avsändare och mottagare samt MAC-adress för avsändare och mottagare. Se tabell 2 för exempel. Om flera alternativa vägar existerar behöver du bara redogöra för en av de möjliga vägarna. (2 B-poäng)

Steg i paketets väg:	**Switch A**	**Router C**	**[...]**	**HTTP-server**
IP-adress Avsändare:	10.1.1.1	?	[...]	?
IP-adress Mottagare:	?	?	[...]	?
MAC-adress Avsändare:	KA:1	?	[...]	?
MAC-adress Mottagare:	?	?	[...]	?

Tabell 2: Exempel på tabell för uppgift 6b. (Tabellen ska utökas för alla steg, alla frågetecken ska ersättas med korrekt information.)

Lösning: I tabellen nedan visas hur paketet ser ut efter att varje enhet tagit emot paketet och ska skicka iväg det till nästa enhet på vägen. För HTTP-servern visas hur paketet ser ut när servern tar emot det.

Steg i paketets väg:	**Switch A**	**Router C**	**Router A**	**HTTP-server**
IP-adress Avsändare:	10.1.1.1	10.1.1.1	10.1.1.1	10.1.1.1
IP-adress Mottagare:	10.1.2.1	10.1.2.1	10.1.2.1	10.1.2.1
MAC-adress Avsändare:	KA:1	RC:1	RA:1	RA:1
MAC-adress Mottagare:	RC:3	RA:3	SH:1	SH:1